RafaelReyesCarmona Final v0.5 ✓ History

1 contributor

318 lines (270 sloc) | 9.03 KB

```
/*
TinyTrackGPS.cpp - A simple track GPS to SD card logger.
TinyTrackGPS v0.5

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

/**************************************************************************
/  Programa de localizacion por gps que graba las posiciones en
/  un fichero de texto cada segundo, de forma diaria.
/
```

```
/  - Conectar módulo SD con pin CS (naranja) en pin 10 arduino.
/
/  Uso de librería TinyGPS.
/   Requiere uso de librería SoftwareSerial, se presupone que disponemos
/   de un dispositivo GPS serie de 9600-bauds conectado en pines 9(rx) y 8(tx).
/  - Conectar módulo NMEA-6M (gps) pines 8,9 (9 - pin rx negro)
/
/  - Conectar LCD 16x2 pines 2,3,4,5,6,7 (2-amarillo , 3-azul,
/     4-rojo, 5-azul oscuro, 6-verde, 7-blanco)
/
/  - Conectar OLED 0.96" en SDA y SCL. pines A4 y A5 del Arduino UNO.
*******************************************************************************/

// Include libraries.
#include <Arduino.h>
#include "config.h"
#include "Display.h"
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SdFat.h>
#include <sdios.h>
#include "UTMconversion.h"

// Variables para grabar en SD.
char GPSLogFile[] = "YYYYMMDD.csv"; // Formato de nombre de fichero. YYYY-Año, MM-Mes, DD-Día.
const uint8_t CHIP_SELECT = SS;  // SD card chip select pin. (10)
SdFat card;   //SdFat.h library.
SdFile file;
boolean SDReady;
boolean SaveOK;

// Variables y clases para obtener datos del GPS y conversion UTM.
TinyGPS gps;
GPS_UTM utm;
SoftwareSerial gps_serial(9, 8);
int year_actual;
byte month_actual, day_actual;
byte hour_prev, minute_prev, second_prev;
float flat, flon;
int year;
byte month, day, hour, minute, second, hundredths;
unsigned long age;
int elev;
int sats;

// Definimos el Display
#if defined(DISPLAY_TYPE_LCD_16X2)
Display LCD(LCD_16X2);
#elif defined (DISPLAY_TYPE_SDD1306_128X64)
```

```
Display LCD(SDD1306_128X64);
#endif

//------------------------------------------------------------------------------
/*
 * User provided date time callback function.
 * See SdFile::dateTimeCallback() for usage.
 */
void dateTime(uint16_t* date, uint16_t* time) {
  // User gets date and time from GPS or real-time
  // clock in real callback function

  // return date using FAT_DATE macro to format fields
  //*date = FAT_DATE(year, month, day);
  *date = (year-1980) << 9 | month << 5 | day;

  // return time using FAT_TIME macro to format fields
  //*time = FAT_TIME(hour, minute, second);
  *time = hour << 11 | minute << 5 | second >> 1;
}
//------------------------------------------------------------------------------

void GPSData(TinyGPS &gps, GPS_UTM &utm);
void ScreenPrint(Display &LCD, TinyGPS &gps, GPS_UTM &utm);
void GPSRefresh();
bool pinswitch();

unsigned long iteration = 0;

void setup(void) {
  //Serial.begin(9600);
  gps_serial.begin(9600);

  //Serial.print(F("Initializing SD card..."));

  SDReady = card.begin(CHIP_SELECT);
  //(SDReady) ? Serial.println(F("Done.")) : Serial.println(F("FAILED!"));

  /* Declaramos pin para selector visor coordenadas */
  //pinMode(PIN_SELECT,INPUT_PULLUP); <--- Produce fallos en el proceso ScreenPrint.
  //pinMode(PIN_SELECT, INPUT); -> Chane every 4 seconds automatically.
  //attachInterrupt(digitalPinToInterrupt(PIN_SELECT), pinswitch, CHANGE); <- Solo pines 2,3. Ardu

  /* Iniciaización del display LCD u OLED */
  LCD.start();
  LCD.clr();
  LCD.splash(750);      // Dibujamos la presensación.

  //Serial.print(F("Waiting for GPS signal..."));
```

```
  LCD.clr();
  LCD.print("Waiting for","GPS signal...");

  unsigned int time = 0;
  bool config = false;

  do {
    LCD.wait_anin(time++);

    for (unsigned long start = millis(); millis() - start < 1000;) {
      while (gps_serial.available()) {
        char c = gps_serial.read();
        //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
        if (gps.encode(c)) {// Did a new valid sentence come in?
          gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
          (age != TinyGPS::GPS_INVALID_AGE) ? config = true : config = false;
        }
      }
    }
  }while(!config);

  sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month, day);

  year_actual = year;
  month_actual = month;
  day_actual = day;
  hour_prev = hour;
  minute_prev = minute;
  second_prev = second;

  //Serial.println(F("Done."));
  //Serial.println(F("Configuration ended."));

  LCD.clr();
}

void loop(void) {
  bool gps_ok = false;
  for (unsigned long start = millis(); millis() - start < 940;) {
    while (gps_serial.available()) {
      if (gps.encode(gps_serial.read())) {
        gps_ok = true;
      }
    }
  }
  iteration++;
  if (gps_ok) {
    GPSData(gps, utm);
    ScreenPrint(LCD, gps, utm);
```

```
  }
}

void GPSData(TinyGPS &gps, GPS_UTM &utm) {
  float f_elevation;
  char utmstr[]= "30S 123456 1234567";
  char timestr[]= "00:00:00";

  gps.f_get_position(&flat, &flon, &age);
  GPSRefresh();

  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
  GPSRefresh();

  f_elevation = gps.f_altitude();
  elev = abs((int)f_elevation);
  //elevation = (int)gps.altitude()/100;
  GPSRefresh();

  sats = gps.satellites();
  GPSRefresh();

  if (utm.UTM(flat, flon)) {
    sprintf(utmstr, "%02d%c %ld %ld", utm.zone(), utm.band(), utm.X(), utm.Y());
  }

  if (age != TinyGPS::GPS_INVALID_AGE){
    sprintf(timestr, "%02d:%02d:%02d,", hour, minute, second);
    //Serial.print(timestr);
  }

  if (year != year_actual || month != month_actual || day != day_actual) {
    sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month, day);
    year_actual = year;
    month_actual = month;
    day_actual = day;
  }

  SdFile::dateTimeCallback(dateTime);

  // Si no existe el fichero lo crea y añade las cabeceras.
    if (SDReady && !card.exists(GPSLogFile)) {
      if (file.open(GPSLogFile, O_CREAT | O_APPEND | O_WRITE)) {
        //Serial.print(F("New GPSLogFile, adding heads..."));
        file.println(F("Time,latitude,longitude,alt,utm"));
        //Serial.println(F("Done."));
        file.close();
        }
        //else {
```

```
        //Serial.println(F("** Error creating GPSLogFile. **"));
        //}
    }

  if (!((hour_prev == hour) && (minute_prev == minute) && (second_prev == second))) {
  if (SDReady && file.open(GPSLogFile, O_APPEND | O_WRITE)) {
    //Serial.print(F("Open GPSLogFile to write..."));
    SaveOK = true;
    char comma = 0X2c;
    file.print(timestr);
    file.print(flat,6);
    file.print(comma);
    file.print(flon,6);
    file.print(comma);
    file.print(elev);
    file.print(comma);
    file.print(utmstr);
    file.print("\n");
    file.close();
    //Serial.println(F("Done."));
    hour_prev = hour;
    minute_prev = minute;
    second_prev = second;
  } else {
    SaveOK = false;
    //Serial.println(F("** Error opening GPSLogFile. **"));
    }
  } //else Serial.println(F("** GPS signal lost. **"));
}

void ScreenPrint(Display &LCD, TinyGPS &gps, GPS_UTM &utm){
/*
  if (pin != digitalRead(PIN_SELECT)) {
    LCD.clr();
    pin = digitalRead(PIN_SELECT);
  }
*/
  if (!pinswitch()) {
        char line[12];

        sprintf(line, "%02d%c %ld ", utm.zone(), utm.band(), utm.X());
        //Serial.println(line);
        LCD.print(0,0,line);
        LCD.print_PChar((byte)6);
        sprintf(line, "%02d ", sats);
        //Serial.println(line);
        LCD.print(12,0,line);
        SaveOK ? LCD.print_PChar((byte)7) : LCD.print("-");

```

```
        // New line
        sprintf(line, "%ld ", utm.Y());
        //Serial.println(line);
        LCD.print(1,1,line);
        LCD.print_PChar((byte)5);
        LCD.print(10,1,"     ");
        sprintf(line, "%dm", elev);
        //Serial.println(line);

        if (elev < 10) LCD.print(14,1,line);
        else if (elev < 100) LCD.print(13,1,line);
        else if (elev < 1000) LCD.print(12,1,line);
        else LCD.print(11,1,line);

  }
  else {
    char line[11];
    LCD.print(1,0,"LAT=");
    dtostrf(flat, 10, 6, line);
    LCD.print(line);
    LCD.print(1,1,"LON=");
    dtostrf(flon, 10, 6, line);
    LCD.print(line);
  }
}

void GPSRefresh()
{
    while (gps_serial.available())
      gps.encode(gps_serial.read());
}

bool pinswitch()
{
  bool pin;
  pin = bitRead(iteration,3); // Change every 4 seconds.
  //pin = digitalRead(PIN_SELECT);
  //LCD.clr(); -> Too slow clear individual characters.
  LCD.print(0,0," ");
  LCD.print(15,0," ");
  LCD.print(0,1," ");
  LCD.print(15,1," ");
  return pin;
}
```